

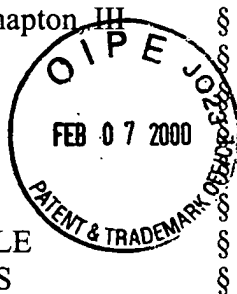
#10
2-22-00
P. 2.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Kenneth S. Knapton III

Group Art Unit: 2762

Serial No.: 09/089,834



Examiner: C. Das

Filed: June 3, 1998

For: BINARY COMPATIBLE
SOFTWARE OBJECTS

Atty. Docket No.: INTL-0033-US

Assistant Commissioner for Patents
Board of Patent Appeals & Interferences
Washington, D.C. 20231

RECEIVED
FEB 18 2000
TC 2700 MAIL ROOM

APPEAL BRIEF

Sir:

Applicant respectfully appeals from the final rejection mailed September 7, 1999

I. REAL PARTY IN INTEREST

The real party in interest is the assignee Intel Corporation.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF THE CLAIMS

Claim 9 was allowed and claims 1-8 and 12-17 are rejected under §§ 112, 102(e) and 103.

IV. STATUS OF AMENDMENTS

All amendments are believed to have been entered but no paper responsive to the paper filed October 27, 1999 was ever received..

02/18/2000 DVUONG 00000045 09089834

01 FC:120

300.00 OP

Date of Deposit: February 2, 2000
I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, DC 20231.
Sherry Tipton
Sherry Tipton

V. SUMMARY OF THE INVENTION

ActiveX controls are a subset of the COM object oriented software technology. COM can use a variety of different object oriented program languages such as C++, Java and Visual Basic. ActiveX controls are typically plugged into a control container which is a type of client. Specification at page 1, lines 5-9.

The ActiveX controls self-register on a computer in a database. In Windows®-based platforms, the database is called a registry. The registry provides a way for a control to advise the client about the control's functionality. More specifically, the ActiveX control places keys in the registry or database to let the container know its functionality. The registry includes information which identifies a particular control or object including Globally Unique Identifiers (GUIDs), Category Identifiers (CATIDs), and Class Identifiers (CLSIDs). Specification at page 1, lines 10-19.

A layer class, wrapper or interface definition is a source code level version of a COM object. It provides an interface between the container or client and the object which may be an ActiveX control. Additional controls may be inserted, snapped in or "plugged in" to a container that already has one or more controls. A plug-in control is source compatible if a new version of the control works unchanged in a container application but the user program must be rebuilt. That is, the application program must be recompiled and then the application can be run without further change. Specification at page 1, lines 20-30.

With a "binary compatible" control, a new version can be plugged into an existing application that was designed and built for the old version. However, the conventional wisdom in the field is that the plug-in must appear to the container as if it were the old version in order for the plug-in to be binary compatible. That is, the plug-in must support the old CLSID and all

interfaces exactly as they were (that is, with the same IIDs, names, dispids, parameters and so forth). See Denning, Adam, "ActiveX Controls Inside Out", Microsoft Press (1997), p. 131. Thus, the conventional wisdom holds that in order to be binary compatible, the same identifiers and interfaces must be used for the plug-in. Specification at page 1, line 21 to page 2, line 12.

A GUID is conventionally hard coded into a layer class. Other objects can then be used with a given container; however, they must have the same interface and GUID in order to work with the layer class in a binary compatible fashion. Specification at page 2, lines 13-17.

Referring to Fig. 1, a method for object oriented programming may be implemented in any object oriented programming technology including COM, ActiveX, Java, Visual Basic, C++ and the like. As indicated in block 10, a first object with a first GUID is registered on the system. Then a second object with a second GUID may be registered as indicated in block 12. Time may pass between the first and second registrations. The registration may be done in any conventional database for registering objects including the so-called registry utilized in Windows®-based platforms. Specification at page 3, lines 10-19.

As indicated in block 14, it is thereafter possible to selectively access one of the first and second objects using the same client or container despite the fact that the objects have different GUIDs. In addition, these first and second objects may be accessed using the same container or client despite the fact that they have different interfaces, for example, in connection with COM based objects. Of course, the dispids and parameters for the two objects still would be the same. However, the first object or the second object may be selectively accessed after snapping in the second object. Specification at page 3, lines 20-30.

In this way, a truly binary compatible object system may be developed in which a second object may be snapped into a container or client containing a first object and the second object

may be utilized without recompiling. No recompiling is necessary even though the first and second objects have different GUIDs and even if the first and second objects have different interfaces, for example, in the case of COM applications. Specification at page 3, line 31 to page 4, line 7.

While these techniques are applicable to a variety of technologies, they are particularly applicable to ActiveX controls wherein first and second controls may be snapped into a container. In this way the container can selectively access either the first or the second object without recompiling. In applications using a layer class or wrapper, the layer class may then be associated with more than one object. A new ActiveX control can be snapped in without creating a new layer class each time, which would require recompiling. Specification at page 4, lines 8-17.

Referring now to Fig. 3, a pair of objects 32 and 34 have different GUIDs and different interfaces. The interface for the object 32 is indicated as I1 and the GUID for object 32 is indicated as GUIDA. Similarly, the GUID for object 34 is GUIDB and the interface for object 34 is I2. Specification at page 6, lines 8-13.

The compiled application, indicated within the dotted line box 36, includes, in the illustrated embodiment, a layer class or wrapper 38 and the client or container 46. As indicated, the layer class communicates with the client 46. Specification at page 6, lines 14-18.

The layer class 38 includes the SetGUID method 42 and the GetGUID method 40. The layer class 38 also stores the selected GUID 44. In the illustrated embodiment, the selected GUID is the GUIDA as indicated by the arrow between the layer class and the object 32. However, the layer class can obtain the GUID for the object 34 through the client from the system database 48. The system database or registry includes GUIDs 50 and 52 in the illustrated

embodiment. Thus, the client can obtain GUIDs for desired objects from the system database 48 and the layer class can access the corresponding object without requiring recompiling. Specification at page 6, lines 19-30. More details on the operation of one embodiment of the invention are set forth in the specification at page 4, line 18 through page 6, line 7.

In prior systems, the objects 32 and 34 would need to have the same GUIDs and interfaces to enable a binarily compatible snap-in system. Now snap-in objects with different GUIDs and different interfaces may be used to implement new functions as necessary in the future without recompiling. In this way, a layer class may be created that has selectively programmable GUIDs for more than one object. Specification at page 6, line 31 to page 7, line 6.

With embodiments of the present invention, one can extend a current technology by allowing updates and new functions. That is, a new object may be used in place of its predecessor. The new object may also be used selectively in conjunction with its predecessor. Specification page 7, lines 7-11.

VI. ISSUES

A. **Can Christensen, Which Does Not Teach Using Different Identifiers for Each of Two Objects, Anticipate Claims 1-5 and 12-13?**

Claims 1-5 and 12-17 call for using different identifiers for each of two objects. The reference does not teach this element and still the claims were rejected under § 102.

B. **Can Christensen, Which Does Not Teach Accessing One of Two Objects in Place of the Other Without Recompiling, Anticipate Claims 6-8?**

Claims 6-8 call for accessing one of two objects in place of the other without recompiling. The reference teaches no such capability and yet the claims are rejected under § 102.

VII. GROUPING OF THE CLAIMS

Claims 1-5 and 12-17 may be grouped. Also, claims 6-8 may be grouped. The patentability of each group is discussed below.

VIII. ARGUMENT

A. **Can Christensen, Which Does Not Teach Using Different Identifiers for Each of Two Objects, Anticipate Claims 1-5 and 12-13?**

Claim 1 was rejected under § 102(e) over Christensen. Claim 1 calls for a method of creating a first object having a first identifier, the first object associated with a first client. A second object is inserted having a second identifier, the second object associated with the first client and the first and second identifiers being different. Thus, the identifiers, which could be GUIDs, are different and yet create a binary compatible structure. This is what the Denning book by Microsoft Press says is simply not possible.

Nothing in the cited Christensen reference does anything to counteract the well established dogma set forth in Microsoft's own book. All Christensen is doing is creating a remote access protocol wherein references on one system to an object on another system can be passed unchanged between the two systems. The way Christensen does this is to make sure that the registries on both systems recognize and have recorded therein the GUID for those objects. There is no plugging in a replacement.

This is absolutely clear from the specification of the Christensen patent. For example, looking at the top of column 8, the client application which corresponds to the left portion in Figure 4 has an object class represented by a unique GUID. See col. 7, line 67. The GUID for that object is set forth in the second line of the operation system registry code set forth in column 8, line 5. The GUID subkey is {42754850-16b7-11ce-90eb-00aa003d7352}. Not surprisingly,

the GUID “after modification by remote automation” is unchanged; as set forth in column 8, line 26, the GUID is as follows: {42754850-16b7-11ce-90eb-00aa003d7352}.

The reason the GUID is unchanged is that both applications, both the client application on the left in Figure 4 and the remote application on the right in Figure 4, have stored in their database or registry, the same GUID. Thus, one can reference a method 64 on the server application from the client application using, on the client computer, the GUID for that method. A reference to the GUID is transferred over the RA channel through the RA object manager and ultimately to the server application.

The GUID or identifier is not changed as called for in claim 1 (“said first and second identifiers being different”). The Christensen reference merely uses the same object on two different systems but never snaps in one object for another object. Thus, the same object is referenced on two different systems using the same GUID. Therefore, claim 1 which calls for using different identifiers patentably distinguishes over the Christensen reference. This is very clearly set forth in column 11 of Christensen:

The Remote Automation application ensures that objects are uniquely represented and identifiable when passed from one computer to another by assigning every object a unique GUID when it is created. GUIDs were explained in detail above. The GUIDs generated are stored in a data structure associated with the RA proxy on the client and a corresponding data structure associated with all RA remote stubs to facilitate lookup by GUID while passing remote object references. Thus, the client and remote server computers all understand and can identify any remote object reference by looking up its GUID in their respective data structures.

Thus, there is no changing of identifiers. The same identifier is used for the same object on both systems in Christensen. The reference to the object can be passed from one system to the other, because both systems include databases that store the same GUID for the object and so

both systems know what the other system is talking about when a given reference is passed between them.

Again referring to claim 1, Christensen does not teach the “first and second identifiers being different” as set forth in claim 1. Therefore, the rejection of claims 1-5 should be overturned.

Claim 12 is a Beauregard style claim that is similar in some respects to claim 1. In particular, it calls for the first and second identifier being different and therefore, for the reasons already discussed, it is respectfully submitted that the rejection of claim 12 and claims 13-17, dependent thereon, should be reversed.

B. Can Christensen, Which Does Not Teach Accessing One of Two Objects in Place of the Other Without Recompiling, Anticipate Claims 6-8?

Claim 6 calls for registering a first object with a first globally unique identifier and registering a second object with a second globally unique identifier. The claim further calls for selectively accessing one of said first and second objects without recompiling.

Christensen teaches using the same GUID for the very reason pointed in Microsoft’s Denning book. Even though the method may be accessed on one of two computers, one of which is remote relative to the other, the same GUID is utilized to always access the given method.

Thus, Christensen does not teach using two different identifiers and selectively accessing one of the first or second objects without recompiling. Therefore, the rejection of claims 6-8 should be reversed.

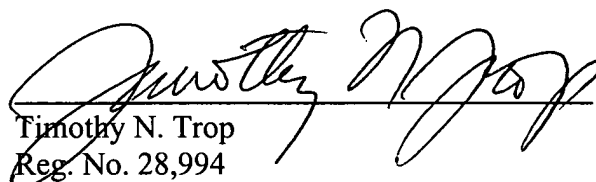
IX. CONCLUSION

Since the rejections of the claims are baseless, they should be reversed.

Respectfully submitted,

Date: _____

2/2/00



Timothy N. Trop

Reg. No. 28,994

TROP, PRUNER, HU & MILES, P.C.

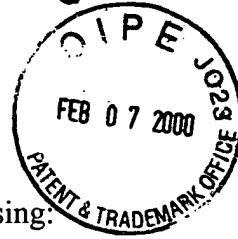
8554 Katy Freeway, Suite 100

Houston, TX 77024

713/468-8880 [Ph]

713/468-8883 [Fax]

APPENDIX OF CLAIMS



The claims on appeal are:

1. A method for object oriented programming comprising:
creating a first object having a first identifier, said object associated with a first client;
inserting a second object having a second identifier, said second object associated with the first client, said first and second identifiers being different; and
using said second object with said first client in place of the first object without recompiling
2. The method of claim 1 including creating a first COM object having a first globally unique identifier, said first COM object associated with a first container, inserting a second COM object having a second globally unique identifier, said second COM object associated with the first container, the first and second globally unique identifiers being different, and using said second COM object with the first container without recompiling.
3. The method of claim 2 including providing a layer class and setting said globally unique identifier in said layer class.
4. The method of claim 1 including creating a layer class that interfaces with one of a plurality of globally unique identifiers of objects associated with said layer class.
5. The method of claim 1 including using said first object again with said first client in place of said second object without recompiling.
6. A method for object oriented programming comprising:
registering a first object with a first globally unique identifier;
registering a second object with a second globally unique identifier; and
selectively accessing one of said first and second objects in place of one another without recompiling.

RECEIVED
FEB 18 2000
TC 1700 MAIL ROOM

7. The method of claim 6 including creating a source code version of said objects, and programming said globally unique identifiers into a layer class.

8. The method of claim 7 including getting the globally unique identifier for each object from a database and setting each globally unique identifier in said layer class.

12. A computer readable storage medium for storing a program including instructions for causing a computer to:

create an object having a first identifier, said object associated with a first client;
insert a second object having a second identifier, said second object associated with the first client, said first and second identifiers being different; and
use said second object with said first client in place of said first object without recompiling.

13. The medium of claim 12 wherein said objects are COM objects.

14. The medium of claim 13 wherein said COM objects are ActiveX controls.

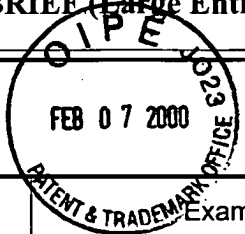
15. The medium of claim 13 wherein said identifiers are globally unique identifiers.

16. The medium of claim 15 including one or more instructions for storing a program instructions for causing a computer to create a layer class having selectively programmable globally unique identifiers for more than one object.

17. The medium of claim 16 including instructions for causing a computer to obtain globally unique identifiers and setting the identifiers in the layer class.

AF/GP 2762

TRANSMITTAL OF APPEAL BRIEF (Large Entity)	Docket No. INTL-0033-US
---	----------------------------



In Re Application Of: KENNETH S. KNAPTON, III	#10 2-22-00 P.2.
--	------------------------

Serial No. 09/089,834	Filing Date June 3, 1998	Examiner C. Das	Group Art Unit 2762
--------------------------	-----------------------------	--------------------	------------------------

Invention: BINARY COMPATIBLE SOFTWARE OBJECTS	RECEIVED FEB 18 2000 C-2700 MAIL ROOM
--	---

TO THE ASSISTANT COMMISSIONER FOR PATENTS:

Transmitted herewith in triplicate is the Appeal Brief in this application, with respect to the Notice of Appeal filed on **December 7, 1999.**

The fee for filing this Appeal Brief is: **\$300.00**

- ☒ A check in the amount of the fee is enclosed.
- ☐ The Commissioner has already been authorized to charge fees in this application to a Deposit Account. A duplicate copy of this sheet is enclosed.
- ☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **20-1504.**
A duplicate copy of this sheet is enclosed.

RECEIVED
FEB 17 2000
OIFE/JCWS


Signature
Timothy N. Trop, Reg. No. 28,994
Trop, Pruner & Hu, P.C.
8554 Katy Freeway, Suite 100
Houston, Texas 77024
Phone: (713) 468-8880
Fax: (713) 468-8883

Dated: February 2, 2000

I certify that this document and fee is being deposited on **February 2, 2000** with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.


Signature of Person Mailing Correspondence

Sherry Tipton
Typed or Printed Name of Person Mailing Correspondence

cc:

TRANSMITTAL OF APPEAL BRIEF (Large Entity)

Docket No.
INTL-0033-US

In Re Application Of: KENNETH S. KNAPTON, III

FEB 07 2000

Serial No.
09/089,834Filing Date
June 3, 1998Examiner
C. DasGroup Art Unit
2762

Invention: BINARY COMPATIBLE SOFTWARE OBJECTS

TO THE ASSISTANT COMMISSIONER FOR PATENTS:

Transmitted herewith in triplicate is the Appeal Brief in this application, with respect to the Notice of Appeal filed on December 7, 1999.

The fee for filing this Appeal Brief is: \$300.00

- ☒ A check in the amount of the fee is enclosed.
- ☐ The Commissioner has already been authorized to charge fees in this application to a Deposit Account. A duplicate copy of this sheet is enclosed.
- ☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 20-1504.
A duplicate copy of this sheet is enclosed.

RECEIVED
FEB 16 2000
TC 2700 MAIL ROOM

Signature

Timothy N. Trop, Reg. No. 28,994
Trop, Pruner & Hu, P.C.
8554 Katy Freeway, Suite 100
Houston, Texas 77024
Phone: (713) 468-8880
Fax: (713) 468-8883

Dated: February 2, 2000

I certify that this document and fee is being deposited on February 2, 2000 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Sherry Tipton

Signature of Person Mailing Correspondence

Sherry Tipton

Typed or Printed Name of Person Mailing Correspondence

CC: